文章编号:1007-6492(1999)03-0104-04

多线程技术在自动报警系统中的应用

尹卫东[1], 张 煊[1], 张建伟[2]

(1.河南省教育信息中心,河南 郑州 450003; 2.河南省商业高等专科学校计算机研究室,河南 郑州 450002)

摘 要: 在自动报警系统中,采用了多线程技术以解决前台监控和后台通讯的协同工作问题.通过对开发工具 Delphi 3.0 中提供的多线程对象 TThread 类,以及临界区、互斥量等同步控制对象类的介绍,阐述了如何编写出简捷、有效的多线程应用程序和安全调用这些程序的方法,并指出在多线程程序开发过程中应注意的事项.

关键词: 多线程; TThread 类; Synchronize 方法; 同步控制对象; 临界区

中图分类号: TP 311 **文献标识码**: B

0 引言

线程(Thread)是多任务操作系统中的一个概念,是一个用于运行代码的 Win 32 对象[1].线程与进程有着许多相似之处,如线程的控制、同步、通信、调度和死锁等问题的解决方法与进程是相同的.但它们又是不同层次的两个独立概念,在 Win 32 中运行代码的是线程而不是进程,进程为在其中运行的线程定义了所需要使用的数据、资源和地址空间.常用的 Windows 95/98 和 Windows NT 均是支持多线程技术的操作系统,目前,多线程已经成为一种独立的技术,获得了越来越广泛的应用.基于 Win 32 的许多编程工具(如 Delphi, C++等)都支持多线程技术,本文以自动报警系统的多线程实现为例,讨论使用 Delphi 3.0 进行多线程编程的方法.

当 Win 32 创建一个新的进程时,会为该进程创建一个线程来运行程序代码,这个线程被称为主线程(Primary Thread).任一线程都可在同一进程中创建一个或多个线程,每一线程都可以执行同一代码,或者不同的线程执行不同的代码段,这些线程均使用同一虚拟地址空间.当主线程结束时,相应的进程从系统中被卸载,标志着一个应用的结束.由于地址空间是由进程提供的,这使得线程之间共享数据变得容易,但同时也增加了对线程管理难度.为方便对多线程的管理,Delphi 提供了专门的工具集.

1 Delphi 中的多线程对象

Delphi 3.0 为多线程编程提供了一个非常实用的多线程对象——TThread 类,利用它可以方便地编写多线程应用程序.创建 TThread 对象的方法是:在 New Items 对话框中选取 Thread Object, Delphi 会显示一个 New Thread Object 对话框,输入新的线程对象类别的名称(如 MyTest),确认后,Delphi 会自动产生如下程序代码:

```
unit Unit1;
interface
uses
  Classes;
type
  MyTest = class(TThread)
  private
    {Private declarations}
  protected
    procedure Execute; override;
  end;
implementation
```

收稿日期:1999-05-06;修订日期:1999-06-30

基金项目:河南省教委科技攻关项目(97520013)

作者简介:尹卫东(1966-),男,河南省西平县人,河南省教育信息中心工程师,主要从事计算机应用研究及开发工作.

Important: Methods and properties of objects in VCL can only be used in a method called using Synchronize. for example,

```
Synchronize(UpdateCaption);
```

and UpdateCaption could look like,

```
procedure MyTest.UpdateCaption;
begin
    Form1.Caption := 'Updated in a thread';
end;
```

```
{ MyTest }
procedure MyTest.Execute;
begin
    {Place thread code here}
end;
end.
```

这段程序代码中有一个过程 Execute，它是该线程的进入点，也就是该线程开始运行时立刻执行的程序。当一个线程被创建立后，会出现两种不同的情况，第一种情况是此线程被立即执行，即运行过程 Execute 中的程序代码；另一种情况是线程创建后即被挂起，直到有执行该线程的请求产生，才会去执行过程 Execute。

线程被创建后处于何种状态，可以通过对构造函数的参数赋值加以控制。TThread 类的构造函数原型如下：

```
constructor Create(CreateSuspended : Boolean);
```

根据接受的参数来指明该线程是否在建立时即被执行。自定义的线程类可通过继承 TThread 类的构造函数，在其中初始化自己的数据。如果要建立一个立即执行的线程，则在构造函数中加入如下程序代码：

```
inherited Create(lalse);
```

反之，则加入

```
inherited Create(true);
```

之后可以调用 TThread 类的 Resume 方法来激活该线程。

2　报警系统中多线程的实现实例

自动报警系统是由上位机和下位机构成的分布式系统，下位机主要负责对各报警点报警信号的采集，上位机主要负责对各报警点信息的管理、查询，并实时监听和处理报警点发出的各种信号。显然，上位机的工作是多任务的，从整体上可将其分为前、后台两部分。前台主要以报警点的信息管理、查询为主，同时接收来自后台的信息并进行相应处理；后台负责与下位机的通讯。为了保证上位机的前、后台协同工作，如上位机系统不能因一个或多个警报信号的处理而中止后台监听，对这样的问题就可以用多线程技术加以解决。

这里以自动报警系统中报警声效的线程程序为例，说明在 Delphi 3.0 中利用 TThread 类编写多线程应用程序的方法。

```
unit BJSound {报警声音效果单元};
interface
uses
  Classes, BJLight {报警视觉效果单元};
type
  TBJSound = class(TThread)
  private
  var
    blnStop : boolean;
    {停止布尔变量，用于终止报警声}
  protected
    procedure Execute; override;
    {建立后首先执行的过程}
    procedure Start;
    {自定义的产生报警声的过程}
  public
    constructor Create ;
    {构造函数}
    procedure StopBJ;
    {终止报警}
  end;
implementation
constructor TBJSound.Create ;
begin
  blnStop := false;
  {以上为初始化数据}
  inherited Create(false);
  {该线程建立后立即执行}
end;
procedure TBJSound.Start ;
begin
  Synchronize(FormBJLight.Flash);
  {在控制图屏幕上产生报警点闪烁效果，使用 Synchronize 方法的目的是为了保证 VCL 的安全使用}
  repeat
  ……
```

```
;以上省略程序是驱动声效设备发出警报
声;
    until BlnStop ;
  end;
  procedure TBJSound.Execute;
  begin
    ;该线程创建后要执行的代码;
    start;
  end;
  procedure TBJSound.StopBJ;
  begin
    BlnStop := true ;
  end;
  end.
```

从上述例子可以看出,在控制图上报警点闪烁效果时使用了一种 Synchronize 方法,它使得 VCL(Visual Components Library 可视化构件库)在多线程中的使用是安全的.前面曾提到所有线程都共享其所属进程的内存空间.在其中存放的局部变量是在各线程的栈中建立的,所以对局部变量的访问是安全的.但对全局变量来说,所有线程都有相同的存取权,所以对全局变量的访问是不安全的,由于在 VCL 中使用了一些全局变量和记录数据类型,因此对于多线程直接访问 VCL 也是不安全的.

为保证对 VCL 访问的安全性,最好的办法是使用 Synchronize 方法.实质上 TThread 类中的 Synchronize 方法是对消息队列的调度:主线程唯一负责对屏幕的操作,当一个由 TThread 实例创建的线程要对屏幕操作时,它首先调用方法 Synchronize 来通知主线程,然后由主线程来完成对屏幕操作.

下面通过对 Synchronize 程序代码的分析,具体说明它如何保证 VCL 的安全性.

```
procedure Tthread.Synchronize(Method:TThread-
Method) ;
  begin
    if FmainThreadWaiting then
      raise Ethread.CreatRes(SMainThreadWait-
ing) ;
    FsynchronizeException := nil ;
    Fmethed := Method ;
      SendMessage ( ThreadWindows, CM-EX-
ECPROC,0,Longint(Self)) ;
      if Assigned ( FSynchronizeExcepion) then
raise FsynchronizeException ;
  end;
```

在 Windows 中,SendMessage 这个程序会等到接收信息的窗口完成了发生信息所要进行的工作后,才返回送信息给它的程序,所以,当有数个程序同时传送一个相同的信息给一个窗口时,在操作系统内部就有一个锁定的机制可以让这些程序一次一个的执行.而 ThreadWindow 是 TThread 在建立时所建立的隐含窗口,CM_EXECPROC 则是 VCL 自定义的一个窗口信息,定义如下:

```
const
  CM_EXECPROC = $8FFF ;
```

通过隐含窗口 ThreadWindow 的函数 ThreadWndProc,不难知道,ThreadWindow 在接收到 CM_EXECPROC 信息之后就执行传给 Synchronize 的程序,而对所有不是 CM-EXECPROC 的信息,都交与其它窗口处理.也就是说让隐含窗口执行那些使用了 VCL 的程序,这使得同一时刻只有一个线程可以使用 VCL,从而保证了 VCL 被多线程使用时的安全性.

3 线程的同步机制

Synchronize 方法是 Delphi 为保证 VCL 安全性而使用的一种同步控制方法.实际上,Delphi 为保证多线程程序的安全性提供了多种同步控制的对象类,如临界区(Critical Section)、互斥量(Mutexes)、信号量(Semaphores)、事件(Event)等.所谓同步机制是指用于保证对多个相关线程(进程)在执行顺序上的协调的相应机制.线程由于资源共享和相互合作,产生了两种形式的制约关系:间接相互制约和直接相互制约.为了协调这两种制约,产生了线程同步机[2].

临界区(Critical Section)指线程中一个特定代码段,它不受同一进程中的其它线程的干扰[3].采用临界区是一种最简单和最迅捷的实现多线程顺序存取一个全局变量或数据结构的方法.一个临界区是一个特殊的 Win 32 变量,其允许一个代码段独立的执行而不受其它线程的干扰.

在自动报警系统中,要求上位机对多个报警信号及时予以响应并进行处理,同时发出声、光警报.当上位机收到一个报警信号时,系统会创建一个 BJSound 线程并开始执行.如果有多个报警信号同时到达,系统则会为这些报警信息号各自创建一个发声线程.这样系统会因同时运行了大量的发声线程而变得异常缓慢,显然这种情况是不

允许出现的.于是可将发声代码段做成临界区,当已创建的发声线程正在执行时,不会再去创建另一个发声线程,这样,就避免出现多个发声线程同时运行而造成的阻塞,使得报警信号得以及时响应和处理,从而保证了系统的正常运行.

```
var
  BJS : TRTLCriticalSection;
……
EnterCriticalSection(BJS);
  BJSound.Greate;
  ……
LeaveCriticalSection(BJS);
```

这里的 TRTLCriticalSection 是 Delphi 3.0 自己定义的一个临界区变量记录,在 Windows 95/NT 中,其临界区变量名为 CRITICAL_SECTION.

当然临界区也存在一些缺陷,它们只可在同一进程中使用,而不能跨越到其他进程.另外,临界区没有时间限制,这意味着只有等到在临界区中的线程离开后,其它线程才能进入,否则,后者将会无限制地等下去.对于不同进程的线程之间顺序存取全局数据、或者对等待时间有限制的应用场合,可以使用另一个同步对象互斥量(Mutexes)来弥补临界区的缺陷,但由于其响应速度远慢于临界区,因此只能将它用于那些对速度不十分敏感的地方.

4　小结

综上所述,在 Delphi 中编写多线程应用程序,关键是要考虑以下几点:

(1) 必须同时关注数个独立执行的程序执行单元的状态及其之间的关系.

(2) 提高线程的安全性,防止系统因启动过多的线程而造成拥塞.

(3) 协调好线程之间的同步机制,避免线程之间因争夺资源而导致死锁.

(4) 合理设置线程的优先级,使系统工作效率最高.

参考文献:

[1] MILLER T, POUELL D. Delphi 3 开发使用手册[M]. 赵海燕,张　杰,桑毅宏,等译. 北京:机械工业出版社,1998.

[2] 汤子瀛. 计算机操作系统. 第2版[M]. 西安:西安电子科技大学出版社,1998.

[3] 徐新华. Delphi 3 编程指南[M]. 北京:宇航出版社,1995.

Application of the Technique of Multithreading at the Auto – alarm System

YIN Wei – dong[1], ZHANG Xuan[1], ZHANG Jian – wei[2]

(1. Henan Information Center of Education, Zhengzhou 450003, China; 2. Research lab of Computer Science, Henan Commercial School, Zhengzhou 450002, China)

Abstract: Using the technique of multithreading, we resolve the problem of how to work properly between the foreground control and the background communication in the auto – alarm system. This paper introduces the method of how to use TThread which is one of multithreading object classes and how to use Critical Section and Mutexes which are synchronized control object classes provided in Delphi 3.0. We give a simply and effective method of how to program a multithreading application and how to safely transfer programs. In the end, itpoints out some key issues in the development of multithreading program.

Key words: Multithreading; TThread class; Synchronize method; Synchronized control object; Critical Section